## Azure Blob Storage Operations

By Amy Coughlin

Cancel

Project Guide

# Azure Blob Storage Operations

Imagine you have been tasked with weaning the technical team off of relying on the Azure portal for performing operations on Azure Blob Storage accounts. Eventually, the team wants to take advantage of automation and consistency available through the use of command-line and scripting languages such as Azure CLI and PowerShell.

Your task is to produce a reference library of code designed to operate on blob storage accounts for upload, download, update, and delete. You will also document hints and tips for troubleshooting and verifying configurations that impact code-based operations.

# Objective 1: Set Up Your Coding Environment

1. Open an **Private Browsing** or **Incognito** browser window to avoid conflicts with your personal Microsoft account.
2. Navigate to the Azure portal, using the link provided with your lab, and log in using the **Credentials** provided with your lab.

   You will be at the **Overview** page for the **Resource group** automatically deployed with the lab, showing the two **Storage accounts** that have also been created for you.
3. Store the **Resource group** name, and the **Name** of each Storage account locally, as you will use them later throughout the lab.
   - The Resource group name is in the upper-left, and ends with **azure-blob-storage-operations**.
   - One Storage account name starts with **pslab**.

- The other Storage account name starts with **labshell**.

4. Click on the name of the storage account that starts with **pslab**.
5. In the left-hand menu of the storage account, navigate to **Data storage** > **Containers**.
6. Click on **+ Add container** to add a new container.
7. **Name** the new container `my-container`.
8. Set **Anonymous access level** to **Container**.
9. Click **Create**.
10. Click **+ Add container** again to add another container.
11. Name this one `my-container2`.
12. Again, set **Anonymous access level** to **Container** and click **Create**.
13. Click on the **my-container** container to open it and leave it ready for later use.
14. Launch **Cloud Shell** by clicking the terminal icon in the upper-right of the Azure Portal.
15. Within the panel that opens at the bottom, set up your Cloud Shell as follows:
    1. Click on **Bash**.
    2. Select **Mount storage account**.
    3. Select the only option under **subscription**.
    4. Click **Apply**.
    5. Ensure **Select existing storage account** is selected, and click **Next**.
    6. Under **Resource group** select the only option. (The subscription should be selected for you. Do not try to set up a new resource group.)
    7. For **Storage account name** select the one that starts with **labshell**.
    8. Click the **Create a file share** link, and enter `fileshare`. (Any allowed name would work, as you will not need to use this name later).
    9. Click **Select**.
16. Wait for the shell to initialize.

    It'll take one or two minutes, and once done will leave you at a Bash terminal `cloud [ ~ ]$` prompt

You are now ready for coding.

# Objective 2: Perform CRUD Operations Using the Azure CLI

**Important:** In the code snippets for the rest of this lab, anywhere you see `[your storage account]`, you will need to replace that with the full name of your storage account (this will start with `pslab`). If the command includes double-quotes, so `"`, keep the double-quotes.

1. At the command line, execute this code to retrieve a list of the two storage account keys, primary and secondary:

   ```
   az storage account keys list --account-name [your storage account]
   ```

2. Copy `key1`'s `value`, the primary key, from the returned JSON output (either key is acceptable).

   It'll be a long series of characters.

3. At the command line, execute the following code to export the key as an environment variable

   ```
   export AZURE_STORAGE_KEY="[your key]"
   ```

   NOTE: Replace `[your key]` with the account key you copied in the last step. Make sure the key is still encased in double quotes, and that there is no space before or after the `=` sign.

4. Create a file named `hello.txt` and add some sample content by using this code:

   ```
   echo "Placeholder content via CLI" > hello.txt
   ```

5. Upload the file to `my-container` by executing this code:

   ```
   az storage blob upload \
     --account-name [your storage account] \
     --account-key $AZURE_STORAGE_KEY \
     --container-name my-container \
     --name hello.txt \
     --file hello.txt
   ```

6. Do the following in the portal above the Bash terminal, on the **my-container** page:
   1. Click **Refresh** in your container view.
   2. Click on the newly uploaded **hello.txt** blob.
   3. Click **Edit** to verify the content is **Placeholder content via CLI**.
7. Back at the command line, modify `hello.txt` to include slightly different content using this code:

   ```
   echo "Updated content via CLI" > hello.txt
   ```

8. To perform an update, execute the upload command again:

   ```
   az storage blob upload \
     --account-name [your storage account] \
     --account-key $AZURE_STORAGE_KEY \
     --container-name my-container \
   ```

```
  --name hello.txt \
  --file hello.txt
```

There is no `--overwrite` argument, so this fails with a `The specified blob already exists.` message as expected; you will add that in next.

9. Now, retry the update, including the `--overwrite` argument:

```
az storage blob upload \
  --account-name [your storage account] \
  --account-key $AZURE_STORAGE_KEY \
  --container-name my-container \
  --name hello.txt \
  --file hello.txt
  --overwrite
```

10. In the portal, click **Refresh** near the editing window to verify that the blob has been updated with the **Updated content via CLI** content.

11. At the command line, download the blob locally using this command:

```
az storage blob download \
  --account-name [your storage account] \
  --account-key $AZURE_STORAGE_KEY \
  --container-name my-container \
  --name hello.txt \
  --file downloaded-hello.txt
```

12. Delete the blob using the following command:

```
az storage blob delete \
  --account-name [your storage account] \
  --account-key $AZURE_STORAGE_KEY \
  --container-name my-container \
  --name hello.txt
```

13. In the portal, click **Refresh** near the editing to confirm the blob has been deleted, and that you see the message **The specified blob does not exist.**

14. To confirm your blob download, do the following:

    1. Click **Editor** in the menu bar at the top of the Cloud Shell terminal.
    2. **Confirm** the use of the **Classic** experience.
    3. Click the **{ }** curly brackets icon to open the editor, which includes a file browser.
    4. Verify both your `hello.txt` and `downloaded-hello.txt` files appear in the list.

# Objective 3: Perform CRUD Operations Using PowerShell

1. In the Azure Portal, close the current shell session by clicking the **X** in its upper-right.

   If you're not at the **Containers** page, do one or both of the following

   1. Close the **hello.txt** panel by clicking the **X** in its upper-right.
   2. Then you may need to close the **my-container2** panel to by clicking the **X** in its upper-right.
2. At the **Containers** page, click on **my-container2**.
3. Click the terminal icon at the top of the portal screen to launch **Cloud Shell** again.
4. Once it loads, in the terminal's upper-left click **Switch to PowerShell**, then **Confirm** on the pop-up.
5. At the command line's `PS /home/cloud>` prompt, use the code below to retrieve the primary storage account key (and store it in the `$key` variable).

   ```
   $key = (Get-AzStorageAccountKey -ResourceGroupName "[your resource group]" -Name "[your storage account]")[0].Value
   ```

   Be sure to replace `[your resource group]` and `[your storage account]` with the names in your lab (your storage account name will start with `pslab`). Ensure both are wrapped in double quotes; so keep the `"` in the above command.
6. Execute the code below to create a PowerShell context for interacting with your storage account in subsequent code blocks.

   ```
   $ctx = New-AzStorageContext -StorageAccountName "[your storage account]" -StorageAccountKey $key
   ```

   Be sure to replace `[your storage account]` with the Storage account name in your lab (it will start with `pslab`). Ensure it's wrapped in double quotes; so keep the `"` in the above command.
7. Create a text file using the `Set-Content` command:

   ```
   Set-Content -Path "pshello.txt" -Value "Placeholder content"
   ```
8. To upload the file to my-container2, use the following command:

   ```
   Set-AzStorageBlobContent -File "pshello.txt" -Container "my-container2" -Blob "pshello.txt" -Context $ctx
   ```
9. In the Azure Portal, click **Refresh** in **my-container2** to confirm the upload of **pshello.txt** was successful.
10. Back in the shell, modify the contents of the file using `Set-Content` again:

```
Set-Content -Path "pshello.txt" -Value "Updated content via PowerShell"
```

11. To perform an update, re-upload the file using the same command used for the upload, but this time include the `-Force` argument to allow an overwrite:

    ```
    Set-AzStorageBlobContent -File "pshello.txt" -Container "my-container2" -Blob "pshello.txt" -Context $ctx -Force
    ```

12. Confirm the update by clicking **Refresh** in the portal, then click **pshello.txt**, **Edit**, and note that the contents are **Updated content via PowerShell**.

13. To read the blob by downloading it, run the following command:

    ```
    Get-AzStorageBlobContent -Container "my-container2" -Blob "pshello.txt" -Destination "downloaded-pshello.txt" -Context $ctx
    ```

14. To delete the blob, use the following command:

    ```
    Remove-AzStorageBlob -Container "my-container2" -Blob "pshello.txt" -Context $ctx
    ```

15. In the portal, click **Refresh** near the file editing window to confirm successful deletion, indicated by the message **The specified blob does not exist.**

16. To confirm your blob download, do the following:

    1. Click **Editor** in the menu bar at the top of the Cloud Shell terminal.
    2. **Confirm** the use of the **Classic** experience.
    3. Click the **{ }** curly brackets icon to open the editor, which includes a file browser.
    4. Verify both your `pshello.txt` and `downloaded-pshello.txt` files appear in the list.

Hang tight while we start your Hands-on Lab. Your environment will be ready in approximately 00:14

Creating Lab Environment

## Lab Tools

Lab Diagram

## Learning Objectives

Successfully complete this lab by achieving the following learning objectives.

1. Set Up Your Coding Environment
2. Perform CRUD Operations Using Azure CLI
3. Perform CRUD Operations Using PowerShell